Symbolic Dynamics

Philippe Lemoine, Wisama Khalil

Symbolic Dynamics

Philippe LEMOINE[1] and Wisama KHALIL[2]

Synonyms Explicit dynamics

Definition

The terminology 'Symbolic Dynamics' is used in robotics when the dynamics of robots is described by symbolic expressions using symbolic variables that do not have numerical values. The computation of these expressions must be done using specific software. The symbolic output can constitute a computation program to get the numerical solution after assigning the numerical values for the necessary constants and variables of the problem.

Theory & Application

1. Introduction

The dynamics of robots involves the development of their equations of motion, which describe the relationship between the input joint efforts (forces or torques) and the output motion. In this article, two basic models will be treated: the inverse dynamic model (IDM), and the Direct Dynamic Model (DDM). The IDM is used in control applications: it calculates the input joint efforts to achieve a set of prescribed joint accelerations. The DDM is used in simulation applications: it calculates the

[1] Ph. Lemoine
École Centrale de Nantes
Laboratoire des Sciences du Numérique de Nantes (LS2N), UMR CNRS 6004
1 rue de la Noë, BP92101, 44321 Nantes Cedex 03 – France
e-mail : philippe.lemoine@ls2n.fr

[2] W. Khalil deceased in November 2017 during the redaction of this article. He has been Professor at École Centrale de Nantes, and has done his research work at LS2N

joint accelerations resulting from a set of input joint efforts. The study is focused on tree-structure robots with rigid links. The generalization to robots with rigid links and flexible joints is straightforward using the same technique.

As an application in this article, the IDM of a RRP serial robot will be given.

2. Dynamic modeling algorithms

2.1. General form of the equations of motion

The general form of the dynamic equations of motion of tree structure robot with n joints can be written using Lagrange equations as follows (Angeles 2003; Khalil and Dombre 2002):

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau}$ are (n×1) vectors representing the joint positions, velocities and accelerations and input efforts respectively, $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix of the robot, the vector $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represents the Coriolis, centrifugal and gravity efforts.

The IDM can be obtained directly from equation (1). It will be denoted by:

$$\boldsymbol{\tau} = \boldsymbol{idm}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \tag{2}$$

The DDM is obtained from equation (1) by:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M^{-1}}(\boldsymbol{q})\big(\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})\big) \tag{3}$$

It will be denoted by:

$$\ddot{\boldsymbol{q}} = \boldsymbol{ddm}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}) \tag{4}$$

2.2. Efficient dynamic modeling algorithms

The first works in robot dynamics computation used Lagrangian formulation (Uicker 1969; Kahn 1969), but the proposed model needed more than 100 000 mathematical operations to calculate the IDM of a 6 degrees of freedom (dof) robot. To reduce this cost of calculation, three tools have been used: a) neglecting some

elements to obtain an approximated model, b) developing new efficient algorithms, c) using symbolic software computation which can take into account the particular values of the robot parameters.

A breakthrough result in the development of efficient algorithms for IDM computation was reported by (Luh et al. 1980) who proposed a recursive Newton-Euler formulation for serial kinematic chains. The algorithm consists of two recursive equations: a forward one, starting from the base link toward the end-effector link, which computes the link kinematics, followed by a backward one, starting from the end-effector link toward the base link, which computes the joint generalized forces. The number of mathematical operations of this algorithm using numerical computations are $137n$–22 multiplications and $101n$–11 additions, which is linearly proportional to the number of joints ($O(n)$). It gives the inverse dynamic model of a 6 dof serial robot with 800 multiplications and 595 additions. To obtain the DDM, Walker and Orin (1982) proposed to use equation (3), after calculating the matrix $\boldsymbol{M}(\boldsymbol{q})$ and the vector $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ by the efficient inverse dynamic algorithm of Luh et al (1980).

The calculation cost of $\boldsymbol{M}$ can be considerably reduced by using the composite link inertia matrix, and taking into account the fact that $\boldsymbol{M}$ is symmetric (Walker and Orin 1982).

An efficient solution for the DDM for serial chain is proposed in (Featherstone 1983). The cost of this algorithm using numerical computation is $O(n)$ and takes $300n$–267 multiplications and $279n$–259 additions. For n=6, these figures yield 1533 multiplications and 1415 additions. It provides the DDM without calculating the inverse inertia matrix. This algorithm is based also on Newton-Euler formulation using three recursive equations.

2.3. Generalization of Luh and Featherstone works

Based on the results of (Luh et al. 1980) and (Featherstone 1983), the following developments have been achieved:

- Development of symbolic customized programs in order to take advantage of the special structure of each robot (Khalil and Kleinfinger 1987; Khosla 1986). This leads to reduce the computational cost by a factor around 5.
- Dynamics of tree-structure and closed-loop robots. This work is realized thanks to the proposition of a new method to describe the structure of these systems (Khalil and kleinfinger 1986).
- Operational Space dynamics (Lilly and Orin 1990; Khatib 1987).
- Hybrid dynamic models (Featherstone 2008), where the inverse and direct notions are defined on the joint level. The accelerations of some joints are supposed known and their input efforts must be calculated (inverse problem), whereas the input efforts of the other joints are known and their output accelerations must be calculated (direct problem).
- Dynamics of floating base systems (Featherstone 2008; Khalil et al. 2014).

- Dynamics of robots with flexible joints and flexible links, which is discussed in section 6.1.

The next section is devoted to the symbolic generation of IDM of tree-structure systems. The other models and other system programing can be obtained easily using the same strategy.

3. Automatic generation of the symbolic IDM of tree-structure robots

3.1. Introduction

The automatic generation purpose is to develop a computer program that can provide the dynamic models of robots with efficient, accurate, and error free results. Furthermore, if the model must be implemented on a real system for control purpose or to be called many times in an analysis application, the computational cost of the given model must be reduced. From the overview of the modeling algorithms presented in section 2, it can be deduced that efficient computational algorithms are now available for calculating IDM and DDM for robots with rigid or flexible links. Furthermore, they are purely algebraic and their computation is composed of matrix operations without differentiation. This section discusses how to use the algorithms mentioned in section 2 to obtain an optimized model. To facilitate the presentation, this section will be focused on the IDM of tree-structure robots with rigid links and fixed base. The tree-structure IDM can also be exploited in the modeling of closed-loop robots, whose computation is based on modeling at first a virtual tree structure system (Khalil and Kleinfinger 1987; Nakamura and Ghodoussi 1988; Featherstone 1983) which is then closed using loop-closure constraint equations that can be considered using either Lagrange multipliers or virtual power principle.

3.2. Description of the robot

The tree structure rigid system is described using Khalil and Kleinfinger notations (Khalil and Kleinfinger 1986; Khalil and Dombre 2002). The structure is composed of n+1 links and n joints. Link j is articulated on joint j. The links and joints are numbered such that the numbers increase outward from the base. The topology of the system is defined by the (n×1) vector of precedent links, such that the elements $p(j)$ defines the link precedent to link j. The type of the joint is defined by the

parameter σ_j, which is equal to 0 if joint j is revolute, and equal to 1 if joint j is prismatic. We also have $\overline{\sigma}_j$=1–σ_j. A frame $\mathcal{F}_j$ is attached to each link j such that:

i- $\boldsymbol{z}_j$ is along the axis of joint j;

ii- $\boldsymbol{x}_j$ is along the common normal between $\boldsymbol{z}_j$ and one of the succeeding joint axes. The homogeneous transformation matrix ${}^i\boldsymbol{T}_j$, defining frame $\mathcal{F}_j$ with respect to (wrt) frame $\mathcal{F}_i$, is obtained in general as a function of six geometric parameters $\gamma_j, b_j, \alpha_j, d_j, \theta_j, r_j$ (Khalil and Dombre 2002). However, if $\boldsymbol{x}_i$ is along the common normal between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, both γ_j and b_j will be zero. The matrix ${}^i\boldsymbol{T}_j$ will be denoted as:

$$ {}^i\boldsymbol{T}_j = \begin{bmatrix} {}^i\boldsymbol{R}_j & {}^i\boldsymbol{P}_j \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{5} $$

where ${}^i\boldsymbol{R}_j$ is the (3×3) rotation matrix and ${}^i\boldsymbol{P}_j$ is the translation vector of frame j with respect to frame i.

Note that the definition of $\mathcal{F}_j$ such that its $\boldsymbol{z}_j$ axis is along the axis of joint j, leads to symbolic expressions which are simpler than those obtained using $\boldsymbol{z}_j$ along the joint axis j+1. Taking $\boldsymbol{z}_j$ along the joint axis j allows also to calculate the minimum inertial parameters, using simple rules (Khalil et al. 1989; Gautier and Khalil 1990; Khalil and Bennis 1994).

3.3. Newton-Euler formulation

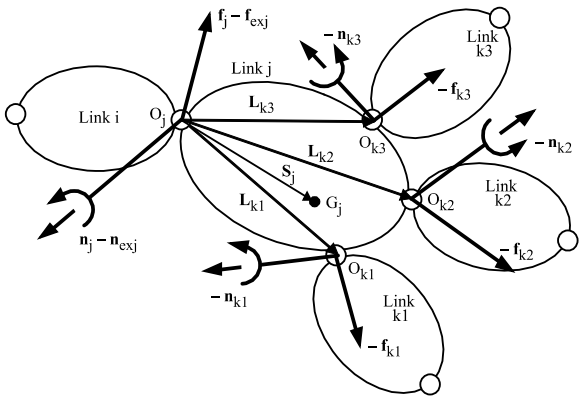

Figure 1. Forces and moments acting on a link of a tree structure

The Newton-Euler formulation describes the dynamics of a body in terms of forces and moments acting on it.

The resultant of total wrenches (forces and moments) on link j can be deduced from Figure 1, as:

$$\boldsymbol{F}_{tj} = \boldsymbol{F}_j - \sum_k \boldsymbol{F}_k - \boldsymbol{F}_{exj} \tag{6}$$

Where $\boldsymbol{F}_{tj} = \left[\boldsymbol{f}_{tj}^T \;\; \boldsymbol{n}_{tj}^T\right]^T$ is the total wrench on the link j, composed of a force $\boldsymbol{f}_{tj}$ and a moment $\boldsymbol{n}_{tj}$, $\boldsymbol{F}_j$ is the reaction wrench on link j due to its precedent link i, where i=$p(j)$, $\boldsymbol{F}_{exj}$ is the supposed known external wrench exerted by link j on the environment, and $\boldsymbol{F}_k$ is the reaction wrench of link j on link k, with $p(k)$=j. If link j is a terminal link, the wrench $\boldsymbol{F}_k$ will be eliminated.

3.4. Inverse dynamic model algorithm

We denote by $\dot{\boldsymbol{V}}_j$ the acceleration of link j, which is composed of the linear acceleration of the origin of frame j, noted $\dot{\boldsymbol{v}}_j$, and the angular acceleration of frame j, noted $\dot{\boldsymbol{\omega}}_j$, such that $\dot{\boldsymbol{V}}_j = \left[\dot{\boldsymbol{v}}_j^T \;\; \dot{\boldsymbol{\omega}}_j^T\right]^T$.

Recall that the inverse dynamic algorithm is composed of two recursive equations:

i- The forward, for j=1 to n, calculates ${}^i\boldsymbol{T}_j$ and ${}^j\boldsymbol{\omega}_j$, ${}^j\dot{\boldsymbol{V}}_j$, ${}^j\boldsymbol{F}_{tj}$ using the following equations (Khalil and Dombre 2002):

$$i = p(j)$$

$${}^j\boldsymbol{\omega}_j = {}^j\boldsymbol{R}_i\,{}^i\boldsymbol{\omega}_i + \overline{\sigma}_j\,{}^j\underline{\boldsymbol{a}}_j\dot{q}_j = {}^j\boldsymbol{\omega}_i + \overline{\sigma}_j\,{}^j\underline{\boldsymbol{a}}_j\dot{q}_j \tag{7}$$

$${}^j\dot{\boldsymbol{V}}_j = {}^j\boldsymbol{S}_i\,{}^i\dot{\boldsymbol{V}}_i + \ddot{q}_j\,{}^j\overline{\boldsymbol{a}}_j + \begin{bmatrix} {}^j\boldsymbol{R}_i\left({}^i\boldsymbol{\omega}_i \times \left({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{P}_j\right)\right) + 2\sigma_j\left({}^j\boldsymbol{\omega}_i \times \dot{q}_j\,{}^j\underline{\boldsymbol{a}}_j\right) \\ \overline{\sigma}_j\left({}^j\boldsymbol{\omega}_i \times \dot{q}_j\,{}^j\underline{\boldsymbol{a}}_j\right) \end{bmatrix} \tag{8}$$

$${}^j\boldsymbol{F}_{tj} = \overline{\boldsymbol{I}}_{oj}\,{}^j\dot{\boldsymbol{V}}_j + \begin{bmatrix} {}^j\boldsymbol{\omega}_j \times \left({}^j\boldsymbol{\omega}_j \times \boldsymbol{ms}_j\right) \\ {}^j\boldsymbol{\omega}_j \times \left(\boldsymbol{I}_{oj}\,{}^j\boldsymbol{\omega}_j\right) \end{bmatrix} \tag{9}$$

with:

$$\boldsymbol{ms}_j = \begin{bmatrix} mx_j & my_j & mz_j \end{bmatrix}^T \ , \ \bar{\boldsymbol{I}}_{oj} = \begin{bmatrix} m_j \mathbf{1}_3 & -\boldsymbol{m}\hat{\boldsymbol{s}}_j \\ \boldsymbol{m}\hat{\boldsymbol{s}}_j & \boldsymbol{I}_{oj} \end{bmatrix} \ , \ \boldsymbol{I}_{oj} = \begin{bmatrix} xx_j & xy_j & xz_j \\ xy_j & yy_j & yz_j \\ xz_j & yz_j & zz_j \end{bmatrix}$$

where:

- the upper-left index on the foregoing vectors indicates the projection frame,
- $i = p(j)$, indicates the link precedent to link j,
- ${}^j\underline{\boldsymbol{a}}_j = [0 \ 0 \ 1]^T$ is the (3×1) unit vector along the axis of joint j,
- ${}^j\boldsymbol{S}_i$ the (6×6) screw transformation matrix of frame i wrt to frame j, its elements can be obtained from ${}^i\boldsymbol{T}_j$:

$${}^j\boldsymbol{S}_i = \begin{bmatrix} {}^j\boldsymbol{R}_i & -{}^j\boldsymbol{R}_i \, {}^i\widehat{\boldsymbol{P}}_j \\ \mathbf{0}_{3x3} & {}^j\boldsymbol{R}_i \end{bmatrix} \tag{10}$$

- ${}^j\overline{\boldsymbol{a}}_j$ is the (6×1) vector given by:

$${}^j\overline{\boldsymbol{a}}_j = \begin{bmatrix} {}^j\underline{\boldsymbol{a}}_j^T \sigma_j & {}^j\underline{\boldsymbol{a}}_j^T \overline{\sigma}_j \end{bmatrix}^T = \begin{bmatrix} 0 \ 0 \ \sigma_j \ 0 \ 0 \ \overline{\sigma}_j \end{bmatrix}^T \tag{11}$$

- $\bar{\boldsymbol{I}}_{oj}$ is the (6×6) spatial inertia matrix of link j. The symbols $m_j, \boldsymbol{ms}_j, \boldsymbol{I}_{oj}$ denote respectively, the mass, the (3×1) first moments vector, and the (3×3) inertia matrix at the origin of the local frame $\mathcal{F}_j$ fixed with link j (Khalil and Dombre 2002). Their 10 components are called the standard inertial parameters of link j.
- $\boldsymbol{m}\hat{\boldsymbol{s}}_j$ defines the (3×3) skew matrix of the vector product associated with the vector $\boldsymbol{ms}_j$.
- $\mathbf{1}_3$ is the (3×3) identity matrix.

These equations are initialized by ${}^0\dot{\boldsymbol{V}}_0 = [-{}^0\boldsymbol{g}^T \quad \mathbf{0}_{1\times3}]^T$. This permits to take into account the gravity forces on all the links.

ii- The backward recursive equations calculate for $j=n$ to 1 (Khalil and Dombre 2002):

$$i = p(j)$$

$${}^j\boldsymbol{F}_j = {}^j\boldsymbol{F}_{tj} + {}^j\boldsymbol{F}_{exj} \tag{12}$$

$$\boldsymbol{\tau}_j = {}^j\overline{\boldsymbol{a}}_j^T \, {}^j\boldsymbol{F}_j \tag{13}$$

If i not equal 0, then calculate:

$$^{i}\boldsymbol{F}_{exi} = {}^{i}\boldsymbol{F}_{exi} + {}^{j}\boldsymbol{S}_{i}^{T}\,{}^{j}\boldsymbol{F}_{j} \tag{14}$$

${}^{j}\boldsymbol{F}_{j}$ is the reaction wrench on link j by link i, ${}^{j}\boldsymbol{F}_{exj}$ is the external wrench applied by link j on the environment, which is supposed to be known.

It should be mentioned that ${}^{i}\boldsymbol{F}_{exi}$ is first initialized to the known external wrench applied by link i on the environment. It is then modified by equation (14) to include the contribution of link j and at last used in equation (12) to continue the process of computation.

Note: Expressing the vectors of link j in the local frame $\mathcal{F}_j$ reduces considerably the cost of the computation, in particular because the inertial parameters $\boldsymbol{ms}_j$ and $\boldsymbol{I}_{oj}$ are constant wrt this frame, and the unit vector along the joint axis j is ${}^{j}\boldsymbol{a}_{j}^{T}$=[0 0 1].

3.5. Input data

In order to generate the IDM, the program requires an input file containing the information required to calculate the desired output model.

An example of such a file describing a RRP serial robot is given in section 3.8. The data needed can be summarized as follows:

a) The numerical values defining the structure: the number of joints is given by the parameter `n`. The type of joints and the topology of the system are given by the lists `sigma` and `p` respectively whose components correspond to the σ_j and $p(j)$ values as defined in section 3.2.
b) The geometric parameters $\gamma_j, b_j, \alpha_j, d_j, \theta_j, r_j$ defining the local frames are represented by the lists `gamma`, `b`, `alpha`, `d`, `theta`, `r`, respectively.
c) The inertia parameters $m_j, mx_j, my_j, mz_j, xx_j, xy_j, xz_j, yy_j, yz_j, zz_j$ are represented by the lists `M`, `MX`, `MY`, `MZ`, `XX`, `XY`, `XZ`, `YY`, `YZ`, `ZZ`.
d) The external efforts components $fx_j, fy_j, fz_j, nx_j, ny_j, nz_j$ are given in the `FX`, `FY`, `FZ`, `CX`, `CY`, `CZ` lists.
e) The gravity vector $\boldsymbol{g} = [gx_0, gy_0, gz_0\,]^T$ is given by the list `G`.
f) The joint velocities and accelerations vectors $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are defined by the lists `QP` and `QDP`, respectively.

In order to simplify the formulation we do not consider here the rotor inertia of actuators and the friction torques/forces. The simplest way to take into account the inertia I_{aj} of the rotor and transmission system of actuator j is to add it to the element M_{jj} of the inertia matrix M of equation (1). The friction torques/forces can be taken into account by adding their expression to the right hand side of equation (1).

Note that some parameters are given symbolically but some others can be given numerically. This helps in noticeably reducing the size of the final generated equations. For example, many geometric parameters have zero values and many angles are typically 0 or $\pm\pi/2$.

3.6. Symbolic Computation

Symbolic computation requires using a computer algebra system in order to obtain the model in terms of symbolic expressions. The symbolic output expressions can be generated automatically to constitute a customized numerical code such as Fortran, Mathematica, MATLAB, C, etc. This program can be used to find the numerical solution of the problem. The input of the symbolic dynamic program of tree-structure robots has been presented in the section 3.5. The values of the number of joints `n`, the topology of the tree-structure defined by `p`, and the type of the joints `sigma` must be given numerically. All the other parameters can be given symbolically (general), or numerically (customized). Some standard symbolic simplification instructions can be added to simplify the expressions. Although the output takes advantage of the particular values of the robot parameters (such as 0 or $\pm\pi/2$), the output expanded expressions will be complicated such that the cost of using them in a numerical program will be considerably greater than using general numerical program. The use of intermediate variables and grouping some inertial parameters together permit to obtain an efficient symbolic program.

The next section presents a simple solution for obtaining an optimized symbolic output.

3.7. Symbolic programing using intermediate variables technique

In Robotran and ARM robotics software, the authors presented complicated methods to select the intermediate variables and to group some constant terms together. This section presents another systematic solution, which is based on the following strategy:

i- Using geometric description method, where the transformation matrices ${}^{i}\boldsymbol{T}_{j}$ are calculated with 4 parameters in most cases, whereas the 6 parameters will be exceptional.

ii- Defining the local frame of link j such that $\boldsymbol{z}_j$ is along the joint axis j.

iii- Using the base inertial parameters (minimum set of inertial parameters), which are calculated from the standard inertial parameters by eliminating some parameters and grouping some others together.

iv- Eliminating redundant calculations appearing in the general algorithm (section 3.4). For instance the upper parts of equations (8) and (9), corresponding to the calculation of ${}^{j}\dot{\boldsymbol{V}}_j$ and ${}^{j}\boldsymbol{F}_{tj}$, contain many redundant calculations which can be eliminated by using the matrices ${}^{i}\boldsymbol{U}_i$ and ${}^{j}\boldsymbol{U}_j$ respectively, where ${}^{i}\boldsymbol{U}_i = {}^{i}\widehat{\boldsymbol{\omega}}_i + {}^{i}\widehat{\boldsymbol{\omega}}_i {}^{i}\widehat{\boldsymbol{\omega}}_i$ should have been calculated during a previous iteration. Also some elements needed to calculate ${}^{j}\boldsymbol{n}_j$ should have been calculated while computing ${}^{j}\boldsymbol{U}_j$.

The use of Khalil and Kleinfinger notations verify the first two conditions. These notations allow also the definition of the base parameters using closed-form rules. For instance, if joint j is revolute, the parameters yy_j, mz_j and m_j can be grouped on the inertial parameters of link $p(j)$, whereas if joint j is prismatic, the parameters $xx_j, xy_j, xz_j, yy_j, yz_j$ and zz_j can be grouped on the inertial parameters of link $p(j)$. The grouping relations are general and can be calculated using closed-form relations (Gautier and Khalil 1990).

The use of intermediate variables (Khosla 1986; Khalil and Kleinfinger 1985) is carried out after each computation step (matrix multiplication or addition) by replacing the expressions of the elements of a vector or a matrix by an intermediate variable when this element contains at least one mathematical operation, or composed of `sin` or `cos` functions. The obtained vectors and matrices are propagated in the subsequent equations. This procedure saves the multiplications by one (and minus one) and zero, and additions with zero. At the end, the model is obtained as a set of intermediate variables without incorporating loop computations. The list of output variables is then inspected and the variables that have no effect on the desired output can be eliminated.

The output list can be put in the form of any desired computer code such as MATLAB, Mathematica, Fortran, C, etc.

The computation cost of the inverse dynamic model using this technique for a general robot, whose parameters are symbols without any particular values, is given by $92n$–127 multiplications and $81n$–117 additions, which represent about 55% the cost of a numerical program. This cost will be reduced considerably (to attain 20%) by increasing the number of robot parameters with particular numerical values.

3.8. Example

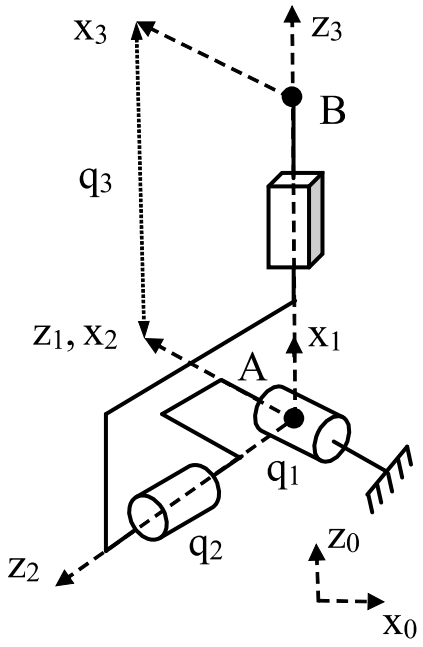

Figure 2. Example of a RRP serial robot

Let us consider the calculation of the IDM of the RRP serial robot given in Figure 2. The input data for the customized symbolic program in given in Figure 3. Note that the geometric parameter corresponding to the joint variable *j* (joint position) is denoted `qj`. This is the case for the parameter `thetaj` when *j* is revolute (`sigma`=0), or for the parameter `rj` when *j* is prismatic (`sigma`=1).

```
(* Name of file : RRP.par *)
(* Numerical values: number of joints, type of joints, topology *)
 n = 3
 sigma = {0,0,1}
 p = {0,1,2}
(* Geometric parameters *)
 b = {b1,0,0}
 d = {d1,0,0}
 r = {r1,0,q3}
 gamma = {g1,0,0}
 alpha = {a1,Pi/2,Pi/2}
 theta = {q1,q2,0}
(* Inertial parameters *)
 XX = {XX1,XX2,XX3}
 XY = {XY1,XY2,XY3}
 XZ = {XZ1,XZ2,XZ3}
 YY = {YY1,YY2,YY3}
 YZ = {YZ1,YZ2,YZ3}
 ZZ = {ZZ1,ZZ2,ZZ3}
 MX = {MX1,MX2,MX3}
 MY = {MY1,MY2,MY3}
 MZ = {MZ1,MZ2,MZ3}
 M = {M1,M2,M3}
(* External forces and torques *)
 FX = {0,0,0}
 FY = {0,0,0}
 FZ = {0,0,0}
 CX = {0,0,0}
 CY = {0,0,0}
 CZ = {0,0,0}
(* Joint velocities and accelerations *)
 QP = {QP1,QP2,QP3}
 QDP = {QDP1,QDP2,QDP3}
(* Acceleration of gravity *)
 G = {0,0,G3}
```

Figure 3. Example of symbolic input file for the RRP serial robot

As already mentioned, in order to reduce the number of operations and improve the efficiency of the symbolic IDM output program, the determination of the base inertial parameters is carried out. The goal of this first step is to get the minimum set of inertial parameters not equal to zero and that are sufficient to calculate the dynamic model of the robot. They are obtained from the classical inertial parameters by eliminating those having no effect on the dynamic model and by regrouping some other parameters. The method that is used is described in (Gautier and Khalil 1990) and (Khalil and Bennis 1994). It consists in applying simple rules that use closed-form relations function of the geometric parameters of the robot. The rules to apply depend on the type of the joints and the topology of the robot, so the base inertial parameters and the calculated grouping relations are specific to each type of robot.

```
Name of file : RRP.regp

Geometric parameters
j  p sigma gamma  b  alpha  d  theta r
1  0   0    g1    b1  a1    d1  q1    r1
2  1   0    0     0  Pi/2   0   q2    0
3  2   1    0     0  Pi/2   0   0     q3

Standard inertial parameters
j  XX   XY   XZ   YY   YZ   ZZ   MX   MY   MZ   M
1  XX1  XY1  XZ1  YY1  YZ1  ZZ1  MX1  MY1  MZ1  M1
2  XX2  XY2  XZ2  YY2  YZ2  ZZ2  MX2  MY2  MZ2  M2
3  XX3  XY3  XZ3  YY3  YZ3  ZZ3  MX3  MY3  MZ3  M3

The regrouping relations are

ZZ1R=YY2 + ZZ1 + ZZ3
MY1R=MY1 - MZ2
XX2R=XX2 + XX3 - YY2 - ZZ3
XY2R=XY2 - XZ3
XZ2R=XY3 + XZ2
YZ2R=YZ2 - YZ3
ZZ2R=YY3 + ZZ2

The base inertial parameters are
j  XX    XY    XZ    YY    YZ    ZZ    MX    MY    MZ    M
1  0     0     0     0     0     ZZ1R  MX1   MY1R  0     0
2  XX2R  XY2R  XZ2R  0     YZ2R  ZZ2R  MX2   MY2   0     0
3  0     0     0     0     0     0     MX3   MY3   MZ3   M3
```

Figure 4. Base inertial parameters and grouping relations of the RRP serial robot

The result of this stage for the RRP serial robot example is given in the output file shown in Figure 4, also containing the corresponding grouping relations. The standard inertial parameters that are eliminated or regrouped on other parameters are put to zero. The parameters on which other ones are regrouped have their name completed with the letter R. The grouping relations are used when replacing the symbolic value of these variables by their numerical value.

A new symbolic input file using the base inertial parameters is also generated and is given in Figure 5. This input file is then used to compute the robot IDM.

A final step, called optimizer, is done after the generation of the symbolic IDM in order to reduce the number of calculations in the output program. At this time one can select the list of the desired output variables (in the example the joint torques are chosen) and the format of the output file (in the example the MATLAB format is chosen). The output program is then inspected and all the variables that are not involved in the computation of the selected output variables are eliminated, implying the elimination of their calculation.

```
(* Name of file : RRPbase.par *)
(* Numerical values: number of joints, type of joints, topology *)
 n = 3
 sigma = {0,0,1}
 p = {0,1,2}
(* Geometric parameters *)
 b = {b1,0,0}
 d = {d1,0,0}
 r = {r1,0,q3}
 gamma = {g1,0,0}
 alpha = {a1,Pi/2,Pi/2}
 theta = {q1,q2,0}
(* Inertial parameters *)
 XX = {0,XX2R,0}
 XY = {0,XY2R,0}
 XZ = {0,XZ2R,0}
 YY = {0,0,0}
 YZ = {0,YZ2R,0}
 ZZ = {ZZ1R,ZZ2R,0}
 MX = {MX1,MX2,MX3}
 MY = {MY1R,MY2,MY3}
 MZ = {0,0,MZ3}
 M = {0,0,M3}
(* External forces and torques *)
 FX = {0,0,0}
 FY = {0,0,0}
 FZ = {0,0,0}
 CX = {0,0,0}
 CY = {0,0,0}
 CZ = {0,0,0}
(* Joint velocities and accelerations *)
 QP = {QP1,QP2,QP3}
 QDP = {QDP1,QDP2,QDP3}
(* Acceleration of gravity *)
 G = {0,0,G3}
```

Figure 5. Symbolic input file using the base inertial parameters of the RRP serial robot

Figure 6 gives the symbolic IDM output file using the base inertial parameters.

Here is the comparison of the calculation costs:

- Numerical (Luh et al.): $101n$–11=292 additions and $137n$–22=389 multiplications.
- Calculation cost using standard inertial parameters: 92 additions and 95 multiplications after optimizer (152 additions and 173 multiplications before optimizer).
- Calculation cost using base inertial parameters: 66 additions and 77 multiplications after optimizer (108 additions and 141 multiplications before optimizer).

The parameters that have been used to compute the IDM are recalled at the beginning of the output file given in Figure 6.

```
% Name of file : RRPbase_dyn.m

% Geometric parameters
% j  p sigma gamma  b  alpha  d  theta r
% 1  0   0    g1    b1  a1    d1  q1   r1
% 2  1   0    0     0  Pi/2   0   q2   0
% 3  2   1    0     0  Pi/2   0   0    q3

% Inertial parameters
% j  XX   XY   XZ   YY   YZ   ZZ   MX   MY   MZ   M
% 1  0    0    0    0    0   ZZ1R  MX1 MY1R  0    0
% 2 XX2R XY2R XZ2R  0   YZ2R ZZ2R  MX2  MY2  0    0
% 3  0    0    0    0    0    0    MX3  MY3  MZ3  M3

% External forces and torques
% j FX FY FZ CX CY CZ
% 1 0  0  0  0  0  0
% 2 0  0  0  0  0  0
% 3 0  0  0  0  0  0

% Joint velocities and accelerations
% j QP   QDP
% 1 QP1  QDP1
% 2 QP2  QDP2
% 3 QP3  QDP3

% Accelerations of gravity : G
%  0
%  0
%  G3

% Dynamic model: Newton Euler method
% Equations:

% Declaration of the function
function RRPbase_dyn()

% Declaration of global input variables
global q1 a1 q2 G3 QDP1 ZZ1R QP1 QP2 QDP2 XX2R
global ZZ2R XY2R XZ2R YZ2R q3 QP3 QDP3 MX3 MY3 MZ3
global M3 MY2 MX2 MY1R MX1

% Declaration of global output variables
global GAM1 GAM2 GAM3
```

Figure 6. Symbolic IDM output file of the RRP serial robot in MATLAB format

```
% Function description:

S1=sin(q1);
C1=cos(q1);
Sa1=sin(a1);
Ca1=cos(a1);
A311=S1.*Sa1;
A321=C1.*Sa1;
S2=sin(q2);
C2=cos(q2);
VP11=-(A311.*G3);
VP21=-(A321.*G3);
VP31=-(Ca1.*G3);
No31=QDP1.*ZZ1R;
WI12=QP1.*S2;
WI22=C2.*QP1;
WP12=QDP1.*S2 + QP2.*WI22;
WP22=C2.*QDP1 - QP2.*WI12;
DV112=-WI12.^2;
DV222=-WI22.^2;
DV332=-QP2.^2;
DV122=WI12.*WI22;
DV132=QP2.*WI12;
DV232=QP2.*WI22;
U122=DV122 - QDP2;
U132=DV132 + WP22;
U212=DV122 + QDP2;
U222=DV112 + DV332;
U232=DV232 - WP12;
U312=DV132 - WP22;
U322=DV232 + WP12;
VP12=C2.*VP11 + S2.*VP31;
VP22=-(S2.*VP11) + C2.*VP31;
PIS22=XX2R - ZZ2R;
No12=WP12.*XX2R - U312.*XY2R + U212.*XZ2R + (-DV222 +
DV332).*YZ2R + DV232.*ZZ2R;
No22=DV132.*PIS22 + U322.*XY2R + (DV112 - DV332).*XZ2R -
U122.*YZ2R;
No32=-(DV122.*XX2R) + (-DV112 + DV222).*XY2R - U232.*XZ2R +
U132.*YZ2R + QDP2.*ZZ2R;
DV113=-WI12.^2;
DV223=-QP2.^2;
DV333=-WI22.^2;
```

Figure 6. Symbolic IDM output file of the RRP serial robot in MATLAB format (continued)

```
DV123=QP2.*WI12;
DV133=-(WI12.*WI22);
DV233=-(QP2.*WI22);
U113=DV223 + DV333;
U123=DV123 + WP22;
U133=DV133 + QDP2;
U213=DV123 - WP22;
U223=DV113 + DV333;
U233=DV233 - WP12;
U313=DV133 - QDP2;
U323=DV233 + WP12;
U333=DV113 + DV223;
VSP13=-(q3.*U122) + VP12;
VSP23=-(q3.*U222) + VP22;
VSP33=-(q3.*U322) - VP21;
VP13=2.*QP2.*QP3 + VSP13;
VP23=VSP33 - 2.*QP3.*WI12;
VP33=QDP3 - VSP23;
F13=MX3.*U113 + MY3.*U123 + MZ3.*U133 + M3.*VP13;
F23=MX3.*U213 + MY3.*U223 + MZ3.*U233 + M3.*VP23;
F33=MX3.*U313 + MY3.*U323 + MZ3.*U333 + M3.*VP33;
N13=-(MZ3.*VP23) + MY3.*VP33;
N23=MZ3.*VP13 - MX3.*VP33;
N33=-(MY3.*VP13) + MX3.*VP23;
N12=N13 + No12 - F23.*q3 - MY2.*VP21;
N22=-N33 + No22 + MX2.*VP21;
N32=N23 + No32 + F13.*q3 - MY2.*VP12 + MX2.*VP22;
N31=C2.*N22 + No31 + N12.*S2 - MY1R.*VP11 + MX1.*VP21;
GAM1=N31;
GAM2=N32;
GAM3=F33;


% *=*
% Number of operations : 66 '+' or '-', 77 '*' or '/'
% The joint torques are given as GAM1,..., GAMn
```

Figure 6. Symbolic IDM output file of the RRP serial robot in MATLAB format (continued)

4. Overview of computer algebra systems for symbolic computation

Many software of computer algebra systems are now available. Some systems are open source, and some others are commercial. They can handle expressions and can carry out matrix operations and trigonometric simplifications. Their use in programing symbolic dynamics makes the programing procedure easy. Among the most cited computer algebra systems, we can mention:

Axiom	open source	http://axiom-developer.org
DoCon	open source	http://www.botik.ru/pub/local/Mechveliani/docon/2.12
Maxima	open source	https://maxima.sourceforge.io
Reduce	open source	https://reduce-algebra.sourceforge.io
SymPy	open source	https://www.sympy.org
Maple	commercial	https://www.maplesoft.com
Mathematica	commercial	https://www.wolfram.com/mathematica
MuPAD	commercial	https://www.mathworks.com/discovery/mupad.html
	See also	https://www.mathworks.com/products/symbolic.html

(MuPAD constitutes the Symbolic Math Toolbox of MATLAB since 2008)

However, some software packages have been developed using general purpose programing languages such as Fortran or C++ after developing their own symbolic manipulation library.

5. Overview of software packages for dynamic models

The emergence of powerful processors and user-friendly languages and software led the scientific community to develop numerical programs able to cover a wide range of applications. Regarding multibody dynamics, several numerical packages were developed. Among those commonly cited are: Adams, Open Dynamics Engine, Simpack, PSpice, Mecano, etc... . Each of them is described as a general-purpose code although, in reality, faced with the huge variety of applications, they all impose some restrictions on the modeling and analysis processes. The symbolic modeling is based on another concept where the generation of the equations is separated from the analysis process. Other than reducing the computational time, the symbolic models can be generated in different languages (Fortran, C, MATLAB, Mathematica,...) and also for different environments for control optimization. They differ in their capabilities in a variety of ways including topologies (serial, tree, closed-loop, mobile), joint models (one or more degrees of freedom, flexible), links model (rigid, flexible), input data, output models (kinematics, dynamics, identification, etc…), and underlying formalism (Newton-Euler, Lagrange), speed of calculation, accuracy, numerical integration routines, integration with other code, application support, user interface, graphics support, and cost.

Table 1 lists some of these software packages:

Software	Main characteristics
Carsim	Focuses on vehicle dynamics
Maplesim	Graphical modeling (Shi and McPhee 2000) Rigid and flexible multibody systems
MOBILE	Object-oriented approach (Kecskemethy 1993; Kecskemethy et al. 1997) Multibody mechatronic systems
MotionGenesis	Produces Fortran programs for simulation
Neweul-M²	Uses the MATLAB environment (Kurz et al. 2010) Mechanical multibody systems
Robotran	Generates kinematic and dynamic models of rigid and flexible multibody systems in MATLAB, Python or C subroutines
ROSAM II	Uses the Maple environment (Kawasaki and Shimizu 1999) Generates kinematic and dynamic symbolic models and calculates the base parameters of serial, tree-structured an closed-loop robots
SD/FAST	Provides symbolic equations as C or Fortran source code
SYMORO+	Uses the (Khalil and Kleinfinger 1986) notations. Generates geometric, kinematic and dynamic models. Calculates the base inertial parameters and generates dynamic identification models. Output models generated in MATLAB, Mathematica, Maple, Fortran or C source code

Table 1. Examples of software packages for symbolic dynamic models

6. Conclusion and further readings

This article has presented the symbolic customized computation of the inverse dynamic model of a RRP serial robot. The model is expressed in terms of intermediate variables.

Inverse and direct dynamic models represent the most important problems in robotics dynamics; however, there are many topics in dynamics that have not been treated. For further readings, the following subjects are suggested:

6.1. Flexible links

In the present article, the joints are supposed perfect and the links are supposed rigid. However, some structures may contain flexibility in the joints or the links that must be taken into account in order to obtain models with acceptable accuracy ap-

proaching the real response of the system. In general, the joint flexibility is modelled using lumped elasticity (Khalil and Gautier 2000; Kruszewski et al. 1975; Wittbrodt et al. 2006) i.e. using 1 dof springs. The link flexibility can be approximated by finite number of lumped springs, but, to obtain a correct model accuracy, a higher number of elements is required, thus increasing the computational time. To have good accuracy, the link-distributed flexibility is treated using finite elements. Some general methodologies based on the Lagrange principle that can be applied to any system are proposed (Book 1984). Some other approaches use the generalized Newton-Euler equations that take into account the flexible variables (Shabana 1990; Sharf and Damaren 1992; Boyer and Khalil 1998). Hybrid dynamic models (inverse and direct problems) for structure with flexible links and joints are also proposed (Khalil et al. 2017).

6.2. The identification of the dynamic parameters

To use the dynamic models in simulation or control, the numerical values of the inertial parameters of the links of the robot are needed. These values can be obtained by identification techniques using an identification model based in expressing the inverse dynamic model as a linear function of the inertial parameters.

An important topic related to this model is the determination of the identifiable parameters. In fact, not all of the inertial parameters can be identified; some of them have no effect and can be eliminated and some others must be grouped together.

The symbolic form of this model is interesting since the identification must be carried out on a very big number of data.

For more details about this topic, the reader can consult (Hollerbach et al. 2016; Khalil and Dombre 2002; Gautier 1991).

Figures

Tables

Table 1. Examples of software packages for symbolic dynamic models

References

Angeles J. (2003) Fundamentals of Robotic Mechanical Systems – Theory, Methods, and Algorithms. Springer-Verlag New York, 2nd edition.

Book W.J. (1984) Recursive Lagrangian dynamics of flexible manipulator arms. The Int. J. of Robotics Research 3(3), pp. 87-101.

Boyer F. and Khalil W. (1998) An efficient calculation of the flexible manipulator inverse dynamics. The International Journal of Robotics Research, 17 (3), pp. 282-293.

Featherstone R. (1983) The calculation of robot dynamics using articulated-body inertias. Int. J. of Robotics Research, Vol. 2(3), pp. 87-101.

Featherstone R. (2008) Rigid Body Dynamics Algorithms. Springer US.

Gautier M. (1991) Numerical calculation of the base inertial parameters. Journal of Robotic Systems, Vol. 8(4), pp. 485-506.

Gautier M. and Khalil W. (1990) Direct calculation of minimum set of inertial parameters of serial robots. IEEE Trans. on Robotics and Automation, Vol. RA-6(3), pp. 368-373.

Hollerbach J., Khalil W. and Gautier M. (2016) Handbook of Robotics, second edition, Chapter 6: Model Identification. Springer Verlag, pp. 113-138.

Kahn M.E. (1969) The near minimum time control of open loop articulated kinematic chains. Ph. D. Thesis, Stanford University, Stanford.

Kawasaki H. and Shimizu T. (1999) Development of Robot Symbolic Analysis System: ROSAM II. Journal of the Robotics Society of Japan, Vol. 17(3), pp 408-415.

Kecskemethy A. (1993) Mobile – An object-oriented tool-set for the efficient modeling of mechatronic systems. Proceedings of the Second Conference on Mechatronics and Robotics, pp. 27-29.

Kecskemethy A., Krupp T. and Hiller M. (1997) Symbolic Processing of Multiloop Mechanism Dynamics Using Closed-Form Kinematics Solutions. Multibody Syst. Dyn., 1, pp. 23-45.

Khalil W. and Bennis F. (1994) Comments on direct calculation of minimum set of inertial parameters of serial robots. IEEE Trans. on Robotics and Automation, Vol. RA-10(1), pp. 78-79.

Khalil W., Bennis F. and Gautier M. (1989) Calculation of the minimum inertial parameters of tree structure robots. Proc. Int. Conf. on Advanced Robotics, Columbus, USA, Springer-Verlag, New York, pp. 189-201.

Khalil W., Boyer F. and Morsli F. (2017) General Dynamic Algorithm for Floating Base Tree Structure Robots With Flexible Joints and Links. Journal of Mechanisms and Robotics 9(3), DOI 10.1115/1.4035798.

Khalil W. and Dombre E. (2002) Modeling, Identification and Control of Robots. Hermes Penton London.

Khalil W. and Gautier M. (2000) Modeling of mechanical systems with lumped elasticity. Proceedings of IEEE International Conference on Robotics and Automation, San Francisco, CA, pp. 3965-3970.

Khalil W. and Kleinfinger J.F. (1985) Une modélisation performante pour la commande dynamique de robots. Revue RAIRO, APII, Vol. 6, pp. 561-574.

Khalil W. and Kleinfinger J.F. (1986) A new geometric notation for open and closed-loop robots. Proceedings of IEEE International Conference on Robotics and Automation, San Francisco, CA, pp. 1174-1180.

Khalil W. and Kleinfinger J.F. (1987) Minimum operations and minimum parameters of the dynamic model of tree structure robots. IEEE Journal of Robotics and Automation, 3 (6), pp. 517-526.

Khalil W., Vijayalingam A., Khomutenko B., Mukhanov I., Lemoine P., Ecorchard G. (2014) Open SYMORO: An open source software package for symbolic modeling of robots. Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron. pp. 126-1211.

Khatib O. (1987) A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE J. of Robotics and Automation RA-3(1), pp. 43-53.

Khosla P.K. (1986) Real-time control and identification of direct drive manipulators. Ph. D. Thesis, Carnegie Mellon University, Pittsburgh, USA.

Kruszewski J., Gawronski W., Wittbrodt E., Najbar F., Grabowski S. (1975) Metoda Sztywnych Elementow Skonczonych (The Rigid Finite Element Method). Arkady, Warszawa.

Kurz T., Eberhard P., Henninger C., Schiehlen W. (2010) From Neweul to Neweul-M²: Symbolical Equations of Motion for Multibody System Analysis and Synthesis. Multibody System Dynamics, Vol. 24, No. 1, pp. 25-41.

Lilly K.W. and Orin D.E. (1990) Efficient O(N) computation of the operational space inertia matrix. Proc. IEEE Int. Conf. on Robotics and Automation, Cincinnati, pp. 1014-1019.

Luh J., Walker M. and Paul R. (1980) On-line computational scheme for mechanical manipulators. ASME Journal of Dynamic Systems, Measurement and Control, 102 (2), pp. 69-76.

Nakamura Y. and Ghodoussi M. (1988) A computational scheme of closed link robot dynamics derived by d'Alembert Principle. IEEE Int. Conf. on robotics and automation, pp. 1354-1360.

Shabana A. (1990) Dynamics of flexible bodies using generalized Newton-Euler equations. Journal of Dynamic Systems, Measurement, and Control, 112, pp. 496-503.

Sharf I. and Damaren C. (1992) Simulation of flexible-link manipulators: basis functions and non-linear terms in the motion equations. Proceedings of IEEE International Conference on Robotics and Automation. Nice, France, pp. 1956-1962.

Shi P. and McPhee J. (2000) Multibody System Dynamics. Kluwer Academic Publishers. https://doi.org/10.1023/A:1009841017268.

Uicker J.J. (1969) Dynamic behaviour of spatial linkages. Trans. of ASME, J. of Engineering for Industry, Vol. 91, pp. 251-258.

Walker M.W. and Orin D.E. (1982) Efficient dynamic computer simulation of robotics mechanism. Trans. of ASME, J. of Dynamic Systems, Measurement, and Control 104, pp. 205-211.

Wittbrodt E., Adamiec-Wojcik I. and Wojciech S. (2006) Dynamics of Flexible Multibody Systems. Rigid Finite Element Method. Springer-Verlag Berlin Heidelberg.

Cross-references

- Kinematics
- Dynamics Calculation Methods
- Recursive Newton-Euler Algorithm
- Dynamics Simulation
- Closed-Loop Dynamics
- Dynamics of Parallel Robots